

Python! Advanced Google Earth Animations from a Feature Class!

Contributed by Kevin Bell
29, Jun. 2011
Last Updated 05, Jul. 2011

Is anyone riding motorcycles from SLC to San Diego for the User Conference? If so, get in touch! (I know a call for riders is a shameless abuse of the GIS Portal…)

Well, if you had a feature class full of points along your route, and fields that explain the type of feature, descriptive text, and a path to images on your web server, then you can make an advanced animation for Google Earth. This isn't super advanced, but it does offer more control than the toy sample that I posted a long time ago.

This will fly to each of your points, and pop up the info window containing your data/picture for a set time, then close it and fly to the next. I've got it set up to view each of our non-compliantly zoned businesses by rotating to look at the store front then showing the detailed shop in a picture with some other information about it. It's kind of fun because it randomly adjusts the tilt angle to give a more dynamic viewing experience.

After launching the resulting KML, the first item in the folder of your placemarks will be a tour icon that says "play me" and that's exactly what you should do.

Be sure that you have a proper spatial reference in your XYZ! Understanding the structure of the resulting KML is important and will help you understand how the code works. After the header you have all of the "fly to" and animation sequence instructions. Within each of these "fly to" specs a placemark is referenced. The placemarks are defined after the "fly to" list.

Happy tour building! (Beware that some of the lines may get word-wrapped.)

```
#-----
#GoogleEarthAnimationFromFeatureClass.py
#Author: Kevin Bell
#       Salt Lake City
#       Information Management Services
#Date: 20110602
#Purpose: write a kml file that will fly to GIS points and animate popup behavior
#-----
import random
import arcpy
fc = r'D:\gis\planning\20110614_googleEarthAnimationFromFC\default.gdb\clean_1'
# slurp up your point data into a dictionary
d = {}
for row in arcpy.SearchCursor(fc):
    d[row.OBJECTID_1] = [row.POINT_Y, row.POINT_X, row.BusName, row.Address, row.BusType, row.ZoningType,
row.Photo, row.heading]
    print row.OBJECTID_1

f = open("TourOfNonConformingProperties2.kml", 'w')
# write the header
f.write('<?xml version="1.0" encoding="UTF-8"?>\n')
f.write('<kml xmlns="http://www.opengis.net/kml/2.2"\n')
f.write('  xmlns:gx="http://www.google.com/kml/ext/2.2">\n')
f.write('\n')
f.write('  <Document>\n')
f.write('    <name>Non-Compliant Properties in SLC</name>\n')
f.write('    <open>1</open>\n')
f.write('    <gx:Tour>\n')
f.write('      <name>Play me</name>\n')
f.write('      <gx:Playlist>\n')

def writeFlyTo(fid, lat, lon, busname, address, bustype, zoning, head):
    """write the anime sequence that references a placemark
    d[i] = [row.POINT_Y, row.POINT_X, row.BusName, row.Address, row.BusType, row.ZoningType]"""
    t = [40, 50, 60, 70] # tilt options
    geTilt = random.sample(t, 1)[0]
```

```

f.write('\n')
f.write('    <gx:FlyTo>\n')
f.write('    <gx:duration>8.0</gx:duration>\n')
f.write('    <gx:flyToMode>smooth</gx:flyToMode>\n')
f.write('    <LookAt>\n')
f.write('        <longitude>%f</longitude>\n' % lon)
f.write('        <latitude>%f</latitude>\n' % lat)
f.write('        <altitude>0</altitude>\n')
f.write('        <heading>%f</heading>\n' % head) ##
f.write('        <tilt>%f</tilt>\n' % geTilt) ##
f.write('        <range>300</range>\n') ##
f.write('    <gx:altitudeMode>clampToGround</gx:altitudeMode>\n')
f.write('    </LookAt>\n')
f.write(' </gx:FlyTo>\n\n')

f.write(' <gx:AnimatedUpdate>\n')
f.write(' <gx:duration>0.0</gx:duration>\n')
f.write(' <Update>\n')
f.write('   <targetHref/>\n')
f.write('   <Change>\n')
f.write('     <Placemark targetId="%s">\n' % fid)
f.write('       <gx:balloonVisibility>1</gx:balloonVisibility>\n')
f.write('     </Placemark>\n')
f.write('   </Change>\n')
f.write(' </Update>\n')
f.write(' </gx:AnimatedUpdate>\n\n')

f.write(' <gx:Wait>\n')
f.write(' <gx:duration>4.0</gx:duration>\n')
f.write(' </gx:Wait>\n\n')

f.write(' <gx:AnimatedUpdate>\n')
f.write(' <gx:duration>0.0</gx:duration>\n')
f.write(' <Update>\n')
f.write('   <targetHref/>,\n')
f.write('   <Change>\n')
f.write('     <Placemark targetId="%s">\n' % fid)
f.write('       <gx:balloonVisibility>0</gx:balloonVisibility>\n')
f.write('     </Placemark>\n')
f.write('   </Change>\n')
f.write(' </Update>\n')
f.write(' </gx:AnimatedUpdate>\n')
def writePlacemark(fid, lat, lon, busname, address, bustype, zoning, photo):
    "d[i] = [row.POINT_Y, row.POINT_X, row.BusName, row.Address, row.BusType, row.ZoningType]"
    f.write(' <Placemark id="%s">\n' % fid)
    f.write('   <name>%s</name>\n' % busname)
    f.write('   <description>\n')
    f.write('     <![CDATA[\n')
    f.write('       %s\n' % address)
    f.write('       %s\n' % bustype)
    f.write('       %s\n' % zoning)
    f.write('     \n' % photo) ##
    f.write('     ]>\n')
    f.write('   </description>\n')
    f.write(' <Point>\n')
    f.write('   <gx:altitudeMode>clampToGround</gx:altitudeMode>\n')##
    f.write('   <coordinates>%f,%f,0</coordinates>\n' % (lon, lat))
    f.write(' </Point>\n')
    f.write(' </Placemark>\n')

for k in d.keys():
    writeFlyTo(k, d[k][0], d[k][1], d[k][2], d[k][3],d[k][4],d[k][5], d[k][7])

```

```
f.write(' </gx:Playlist>\n')  
f.write(' </gx:Tour>\n')
```

```
for k in d.keys():  
    writePlacemark(k, d[k][0], d[k][1], d[k][2], d[k][3],d[k][4],d[k][5],d[k][6])
```

```
f.write(' </Document>\n')  
f.write('</kml>')  
f.close()  
print 'done'
```